LEARN C++
Skill up with our free tutorials

1.1 — Statements and the structure of a program

ALEX[1] FEBRUARY 2, 2024

Chapter introduction

Welcome to the first primary chapter of these C++ tutorials!

In this chapter, we'll take a first look at a number of topics that are essential to every C++ program. Because there are quite a few topics to cover, we'll cover most at a fairly shallow level (just enough to get by). The goal of this chapter is to help you understand how basic C++ programs are constructed. By the end of the chapter, you will be able to write your own simple programs.

In future chapters, we'll revisit the majority of these topics and explore them in more detail. We'll also introduce new concepts that build on top of these.

In order to keep the lesson lengths manageable, topics may be split over several subsequent lessons. If you feel like some important concept isn't covered in a lesson, it's possible that it's covered in the next lesson.

Statements

A computer program is a sequence of instructions that tell the computer what to do. A **statement** is a type of instruction that causes the program to *perform some action*.

Statements are by far the most common type of instruction in a C++ program. This is because they are the smallest independent unit of computation in the C++ language. In that regard, they act much like sentences do in natural language. When we want to convey an idea to another person, we typically write or speak in sentences (not in random words or syllables). In C++, when we want to have our program do something, we typically write statements.

Most (but not all) statements in C++ end in a semicolon. If you see a line that ends in a semicolon, it's probably a statement.

In a high-level language such as C++, a single statement may compile into many machine language instructions.

For advanced readers

There are many different kinds of statements in C++:

- Declaration statements
- Jump statements
- Expression statements
- Compound statements
- Selection statements (conditionals)
- Iteration statements (loops)
- Try blocks

By the time you're through with this tutorial series, you'll understand what all of these are!

Functions and the main function

In C++, statements are typically grouped into units called functions. A **function** is a collection of statements that get executed sequentially (in order, from top to bottom). As you learn to write your own programs, you'll be able to create your own functions and mix and match statements in any way you please (we'll show how in a future lesson).

Rule

Every C++ program must have a special function named **main** (all lower case letters). When the program is run, the statements inside of `main` are executed in sequential order.

Programs typically terminate (finish running) after the last statement inside function `main` has been executed (though programs may abort early in some circumstances, or do some cleanup afterwards).

Functions are typically written to do a specific job or perform some useful action. For example, a function named `max` might contain statements that figures out which of two numbers is larger. A function named `calculateGrade` might calculate a student's grade from a set of test scores. A function named `printEmployee` might print an employee's information to the console. We will talk a lot more about functions soon, as they are the most commonly used organizing tool in a program.

Nomenclature

When discussing functions, it's fairly common shorthand to append a pair of parenthesis to the end of the function's name. For example, if you see the term `main()` or `doSomething()`, this is shorthand for functions named `main` or `doSomething` respectively. This helps differentiate functions from other things with names (such as variables) without having to write the word "function" each time.

In programming, the name of a function (or object, type, template, etc…) is called its **identifier**.

Dissecting Hello world!

Now that you have a brief understanding of what statements and functions are, let’s return to our “Hello world” program and take a high-level look at what each line does in more detail.

```
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "Hello world!";
6      return 0;
7  }
```

Line 1 is a special type of line called a preprocessor directive. This preprocessor directive indicates that we would like to use the contents of the `iostream` library, which is the part of the C++ standard library that allows us to read and write text from/to the console. We need this line in order to use `std::cout` on line 5. Excluding this line would result in a compile error on line 5, as the compiler wouldn’t otherwise know what `std::cout` is.

Line 2 is blank, and is ignored by the compiler. This line exists only to help make the program more readable to humans (by separating the `#include` preprocessor directive and the subsequent parts of the program).

Line 3 tells the compiler that we’re going to write (define) a function whose name (identifier) is `main`. As you learned above, every C++ program must have a `main` function or it will fail to link.

Lines 4 and 7 tell the compiler which lines are part of the *main* function. Everything between the opening curly brace on line 4 and the closing curly brace on line 7 is considered part of the `main` function. This is called the function body.

Line 5 is the first statement within function `main`, and is the first statement that will execute when we run our program. `std::cout` (which stands for “character output”) and the `<<` operator allow us to display information on the console. In this case, we’re displaying the text “Hello world!”. This statement creates the visible output of the program.

Line 6 is a return statement. When an executable program finishes running, the program sends a value back to the operating system in order to indicate whether it ran successfully or not. This particular return statement returns the value `0` to the operating system, which means “everything went okay!”. This is the last statement in the program that executes.

All of the programs we write will follow this general template, or a variation on it.

Author’s note

If parts (or all) of the above explanation are confusing, that’s to be expected at this point. This was just to provide a quick overview. Subsequent lessons will dig into all of the above topics, with plenty of additional explanation and examples.

You can compile and run this program yourself, and you will see that it outputs the following to the console:

```
Hello world!
```

If you run into issues compiling or executing this program, check out lesson 0.8 -- A few common C++ problems (https://www.learncpp.com/cpp-tutorial/a-few-common-cpp-problems/)[3].

Syntax and syntax errors

In English, sentences are constructed according to specific grammatical rules that you probably learned in English class in school. For example, normal sentences end in a period. The rules that govern how sentences are constructed in a language is called **syntax**. If you forget the period and run two sentences together, this is a violation of the English language syntax.

C++ has a syntax too: rules about how your programs must be constructed in order to be considered valid. When you compile your program, the compiler is responsible for making sure your program follows the basic syntax of the C++ language. If you violate a rule, the compiler will complain when you try to compile your program, and issue you a **syntax error**.

Let's see what happens if we omit the semicolon on line 5 of the "Hello world" program, like this:

```
1  #include <iostream>
2
3  int main()
4  {
5     std::cout << "Hello world!"
6     return 0;
7  }
```

Feel free to compile this ill-formed program yourself.

Visual Studio produces the following error (your compiler may generate an error message with different wording):

```
c:\vcprojects\hello.cpp(6): error C2143: syntax error : missing ';' befor
```

The compiler is telling you that it encountered a syntax error on line 6: it was expecting a semicolon before the return statement, but it didn't find one. Although the compiler will tell you which line of code it was compiling when it encountered the syntax error, the thing that needs to be fixed may actually be on a previous line. In this case, we'd conventionally place the semicolon at the end of line 5.

Syntax errors are common when writing a program. Fortunately, they're typically straightforward to find and fix, as the compiler will generally point you right at them. Compilation of a program will only complete once all syntax errors are resolved.

You can try deleting characters or even whole lines from the "Hello world" program to see different kinds of errors that get generated. Try restoring the missing semicolon at the end of line 5, and then deleting lines 1, 3, or 4 and see what happens.

Quiz time

The following quiz is meant to reinforce your understanding of the material presented above.

Question #1

What is a statement?

Show Solution (javascript:void(0))[4]

Question #2

What is a function?

Show Solution (javascript:void(0))[4]

Question #3

What is the name of the function that all programs must have?

Show Solution (javascript:void(0))[4]

Question #4

When a program is run, where does execution start?

Show Solution (javascript:void(0))[4]

Question #5

What symbol are statements in C++ often ended with?

Show Solution (javascript:void(0))[4]

Question #6

What is a syntax error?

Show Solution (javascript:void(0))[4]

Question #7

What is the C++ Standard Library?

Show Hint (javascript:void(0))[4]

Show Solution (javascript:void(0))[4]

Next lesson
1.2 Comments

6

Back to table of contents

7

Previous lesson
0.12 Configuring your compiler: Choosing a language standard

8

9

B U URL INLINE CODE C++ CODE BLOCK HELP!

Leave a comment...

Name*

Email*

Notify me about replies:

POST COMMENT

Find a mistake? Leave a comment above!

Avatars from https://gravatar.com/[12] are connected to your provided email address.

564 COMMENTS Newest

Divakar
April 5, 2024 4:39 am

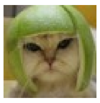
Hello Alex
In question 4 you said execution of program start from main function. Does this work same for compilation
#include <iostream>

```
int sum(int a , int b){
return a+b;
}
int main(){
sum(3,4);
}
```

HERE function main will compile then user defined function sum() will compile ?

Last edited 5 days ago by Divakar

0 Reply



Alex Author

Reply to Divakar 13 April 5, 2024 10:42 am

No. Compilation proceeds top down.

0 Reply

Divakar

Reply to Alex 14 April 5, 2024 10:49 am

Okay thanks Alex

0 Reply

habibi lover

March 26, 2024 3:19 pm

This is AMAZING! I wanna donate

1 Reply

adi barakovic

March 17, 2024 2:41 pm

Amazing site!

2 Reply

I Fold

February 24, 2024 10:42 pm

For the past several weeks, I have been learning Visual Studio 2022. Tonight, after reading your suggestion to change the Properties Configuration from Debug to All Configurations, I noticed a bizarre folder issue. In my Documents folder, I have a Visual Studio 2022 folder, and within that I have a folder labeled My Programs, which contains all my C++ project folders. Within each of those, I usually have two folders (.vc and X64) and several files (.cpp and .sln and three others). But now I suddenly have another folder added with the same name as the project name (e.g., testProject), which has another X64 folder in it, which has another Debug folder in it, and then a shed-load of files (some quite large). The only change I ever make to the default settings is to change the language standard to C++20 for each new project, and I tried changing back to the Debug Properties Setting, but that does not seem to be the issue, so unless this is a result of an update, I cannot see I did anything to have resulted in this extra folder being added to my projects. I am confused. Help!

0 Reply

Alex Author

Reply to I Fold 15 February 27, 2024 6:27 pm

By default, Visual Studio adopts the folder structure `<root>/Solution/Project` -- if your Solution and Project have the same name, you'll get double foldering. If you don't want this, when you create your project you can click the "Place solution and project in the same directory" checkbox.

0 Reply

I Fold

Reply to Alex 16 February 27, 2024 7:52 pm

Now I'm even more confused because I never changed that checkbox; by default, it's always been checked. I never got that "double foldering" before, but now I consistently do. After I posted this query, I realized that there were two other things I did the day before, which was to rename My Programs folder to My C++ Programs (just in case I want to use Visual Studio to learn other programming languages in the future) and I decided to rename most of the projects to better indicate which C++ feature, operator, keyword, etc. that they demonstrate (as opposed to what chapter they came from, since you reorganized and updated your tutorials). To do the later, I used the rename feature in the Solution Explorer drop-down menu, but although it did rename the project, but it did not rename all the files inside that project folder that were associated with the project in my Windows Explorer folders. So, I had a real mess on my hands. I wanted all the files associated with a project to reflect the name change, but since that did not happen, I just decided to delete all my projects and, just in case I missed something important in your updated tutorials, start all over from the very first lesson -- if for no other reason than as a review. I don't see how simply renaming my Windows Explorer folder could cause this "double foldering", so I'm now leaning toward my trying to name my projects, which not only did not rename every file but also, as I recall, added some sub-folders as well. Part of this is my fault for working too late into the night when I'm not as sharp as I could or should be for learning new things. I just hope the "solution" is not to uninstall and then reinstall Visual Studio because half the time I go through that routine I find things do not change because some registry file is not deleted during an uninstall, and then sometimes stubbornly refuses to be manually deleted.

0 Reply

I Fold

Reply to I Fold 17 February 27, 2024 8:32 pm

When I uncheck that box, I do get the "double foldering" you mentioned; however, only the .sln file is in the main folder and everything else is "double foldered". What I am experiencing is slightly different in that the .cpp and .sln files together with three others are in that main folder with everything else "double foldered". Other than what I mentioned above, I don't recall doing anything other than looking through all the Properties Settings just to see what sort of options were available (but the learning curve for Visual Studio is so great that I did not want to mess with anything and cause a problem, which I may have inadvertently done anyway). If I could bother you with one more question, what would be the advantage to having the project and solution separated, albeit still in the same main folder?

0 Reply

Alex Author

Reply to I Fold 18 March 2, 2024 1:08 pm

Having the project and solution separate is preferred when you have a solution with multiple projects.

0 Reply

I Fold

Reply to I Fold 18 February 27, 2024 9:15 pm

Just checked back with Microsoft, and after much searching, it looks as if a few people have been complaining about this very issue since February 20th (apparently after an update). I do not see a way to check for update history, but that timing seems about right, as I always promptly update all my applications. So far, they have not prioritized it, which means it could be a while before the issue is addressed. They always seem to start from the position of denial. Anyway, I just downloaded the newest update (17.9.2), but the problem persists. I just thought I'd let you know.

0 Reply

IceFloe

February 10, 2024 5:52 am

Good evening, I'm back here, and I have a couple of additional questions, if you don't mind.:

1. is the preprocessor directive a set of precompiled code that does not need to be recompiled, but is very important if we want to use some useful commands from it? is it possible to use this directive '#include<iostream>' not only in console applications, but also in game development, where the player is required to enter something or, conversely, it is necessary to display the state of his death on the screen?
2. does this operator include 'a=b+c;'A few machine instructions? after all, the machine adds up the numbers first, and then displays them on the screen?
3. what is the difference between 64-bit and 32-bit processors? Is it that one instruction can contain up to 32 or 64 bits?
4. why is main of type int? because it returns an integer value of 0 or 1? And why would she do that?
5. can the structures be executed not from top to bottom, if, for example, I put int main() from below, and other functions from above? If so, are the other functions and instructions always executed from top to bottom, or can this be changed?

I hope I didn't fill up with questions, it's just very interesting to write here)))

0 Reply

Alex Author
Reply to IceFloe 19 February 11, 2024 10:58 pm

1. No. See https://www.learncpp.com/cpp-tutorial/introduction-to-the-preprocessor/
2. Yes, `a = b + c` should compile down to a few machine instructions, assuming the variables have some fundamental types (e.g. int)
3. The bit-ness of a processor defines how many bits of data can be processed per instruction cycle, as well as how much memory can be addressed.
4. Covered in https://www.learncpp.com/cpp-tutorial/function-return-values-value-returning-functions/
5. Programs don't execute from top to bottom. Ignoring the initialization of global variables, they execute starting at the top of main() and jump to other functions when those functions are called.

1 Reply

christopher
Reply to IceFloe 19 February 11, 2024 10:42 pm

4. because the main function always returns an integer value, its written on this chapter, its this one, "Line 6 is a return statement. When an executable program finishes running, the program sends a value back to the operating system in order to indicate whether it ran successfully or not. This particular return statement returns the value 0 to the

operating system, which means "everything went okay!". This is the last statement in the program that executes.", correct me if I'm wrong ;).

0 Reply

SID
Reply to christopher 20 April 3, 2024 2:49 am

Is it required for the main function to have a return statement of 0 to the operating system so that it says everything is ok or it is possible to rather than have return statement of 0 we go to some other function in its place and also can some other function have the return statement of 0 to say the operating system that everything went fine

0 Reply

S.E.
February 4, 2024 6:09 pm

Took a break, but im back in business. Time to continue this journey.

6 Reply

Atheer Maher
February 1, 2024 3:19 pm

Ok, here is a little mistake:
in the lesson, in order to make a mistake, you wrote:
Let's see what happens if we omit the semicolon on line 5 of the "Hello world" program, like this:

but the error message:
c:\vcprojects\test1.cpp(6): error C2143: syntax error : missing ';' before 'return'

and the subsequent description actually talking about removing the ';' from line 6 not line 5 ...

still, it is excellent lesson and great work and I am really enjoying...

0 Reply

Alex Author
Reply to Atheer Maher 21 February 2, 2024 1:39 pm

It's correct as written. C++ is mostly whitespace agnostic, and does not require the terminating semicolon for a statement to be on the same line as the rest of the statement. Therefore, we could place the semicolon that we'd conventionally put at the end of line 5 on line 6 instead, and it would be valid. Like this:

```
1 #include <iostream>
2
3 int main()
4 {
5    std::cout << "Hello world!"
6    ;return 0; // semicolon unconventionally placed here
7 }
```

It is only when the compiler finds something other than the semicolon it was expecting that it can generate the syntax error -- this happens when processing line 6. Conventionally, we'd fix the issue by adding the semicolon to the end of line 5.

7 Reply

Kamil G.
January 31, 2024 6:29 am

Lesson completed.

I always thought that `cout` and `cin` stand for 'console' output and input, not 'character'.
And I always saw `<<` or `>>` as the direction of what goes where, e.g.:
`console(monitor, screen) << variable/array;` - meaning: variable/array(because string `" "` is a array) goes to the console(monitor) for output.
Or:
`Console(keyboard, mouse (and visually monitor)) >> variables/array;` - meaning: input from keyboard/mouse goes to variable/array.

But I think that 'character' somehow works for it to.

Last edited 2 months ago by Kamil G.

0 Reply

Yossi
January 21, 2024 4:24 am

These lessons are amazing on so many levels. After going through a number of video tutorials (sometimes going 20-30 min deep into them), I eventually decided to look for written tutorials and this suits my learning style to the max! Highly appreciative...

I wanted to ask - you go through the various parts of the statements and what they do, but you don't explain what "std" from "std::cout" does. Standard? What's the purpose of this part of the statement?

0 Reply

Alex Author
Reply to Yossi 22 January 23, 2024 9:07 am

Covered in https://www.learncpp.com/cpp-tutorial/naming-collisions-and-an-introduction-to-namespaces/

Short answer: It tells the compiler to look for `cout` in the scope region named `std`.

2 Reply

Yossi
Reply to Yossi 22 January 21, 2024 4:38 am

Another question (I'm sure this will come up later, but going to ask just in case) - why doesn't "int main()" need a semicolon at the end of it?

0 Reply

Alex Author
Reply to Yossi 23 January 23, 2024 9:08 am

Because the subsequent function body (the curly braces and what's between them) is all part of the function definition of main(). You can put a semicolon after the closing curly brace if you want (though it's not required).

0 Reply

arebenjamin6
January 6, 2024 2:02 pm

is line 1 itself called the preprocessor directive, or is any line with the

```
1 | #include <xxxxxxx>
```

called a preprocessor directive?

0 Reply

Alex Author
Reply to arebenjamin6 24 January 6, 2024 5:42 pm

The `#include <xxxxx>` is the preprocessor directive. It can occur on any line. Line 1 is not special or different in any way.

6 Reply

Arebenjamin6
Reply to Alex [25] January 6, 2024 5:43 pm

Sounds great. I'm loving the tutorial btw! Haven't programmed in years but this makes it interesting. I appreciate all the work you've done here. Have a great day!

1 Reply

Mairigue
January 2, 2024 10:30 am

This is how it spawned I am not able to see a return value, did I forget to add a settings or what im I missing?
#include <iostream>

int main()
{
std::cout << "Hello World!\n";
}

0 Reply

Mairigue
Reply to Mairigue [26] January 2, 2024 11:20 am

I got it to work I just had to manually add "return 0;" and I was able to build it
Is there a way to organically generate it?

Last edited 3 months ago by Mairigue

0 Reply

Alex Author
Reply to Mairigue [27] January 3, 2024 11:04 am

The `return 0` is optional in `main()`, so some IDEs omit it when creating a placeholder program. Just add it yourself if you want it.

0 Reply

OnlyT
December 28, 2023 6:54 am

Is return used like a check to see where the error happened in large code or just something else?

0 Reply

Alex Author
Reply to OnlyT [28] December 28, 2023 2:27 pm

We talk about return and return values here: https://www.learncpp.com/cpp-tutorial/function-return-values-value-returning-functions/

0 Reply

AbeerOrTwo
December 13, 2023 7:11 pm

Hello world!

13 Reply

Hano Sorany
December 12, 2023 4:44 pm

When I copy the code over I get an error telling me to include #include "pch.h"
Is there something wrong with my setup?

0 Reply

Alex Author
Reply to Hano Sorany [29] December 15, 2023 10:35 am

You have precompiled headers turned on. Turn them off.

1 Reply

Fellow-coder
November 24, 2023 5:58 pm

why does this not work? I typed this:
#include <iostream>

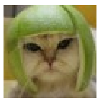
int main()
{
std::cout <<"Hello Merik";
return 0;
}

and it says this:

```
[Running] cd "d:\" && g++ Prac.cpp -o Prac && "d:\"Prac
'g++' is not recognized as an internal or external command,
operable program or batch file.

[Done] exited with code=1 in 0.02 seconds
```

0 Reply


Alex Author
Reply to Fellow-coder [30] November 25, 2023 9:36 pm

Presumably you don't have g++ installed. Try installing it using your package manager.

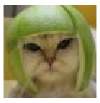
0 Reply

Krishnakumar
November 14, 2023 5:06 am

>In programming, the name of a function (or object, type, template, etc...) is called its identifier.

In C/C++ programming

0 Reply


Alex Author
Reply to Krishnakumar [31] November 14, 2023 9:54 pm

The term identifier is used across many different programming languages.

10 Reply

Carl
November 8, 2023 8:53 pm

I'm learning c++ after a while working in c#. I appreciate what you've done with this site, and look forward to learning from your experience!

I offer a suggestion for a modification to this lesson. In the `nomenclature` section of this lesson, you indicate the parens after a function help to differentiate it from variables etc. I would consider it worth pointing out here that the parens actually serve a significant functional difference, and indicate calling the function. While this concept could be difficult explaining to total beginners, it feels worthwhile planting this seed of a concept.

1 Reply

Alex Author
Reply to Carl 32 November 10, 2023 3:00 pm

This actually one of the things I don't like about the funcName() nomenclature -- it makes it difficult to differentiate whether we're talking about the function itself or a call to the function.

0 Reply

BettoRaite
Reply to Carl 32 November 9, 2023 9:10 am

In c++ you might have a function prototype, or declaration, essentially before any call to a function, you must show the compiler that there is in fact such a function. So, func() doesn't always mean we call it.

1 Reply

Mackenzie
October 22, 2023 5:09 pm

I know this might be answered later but when I omitted line 4({). I noticed that visual studio corrected it by placing { on line 3 which resulted in line 3 being int main() { and line 4 being empty. I ran the program with no errors. So I was just wondering if placing the { on a seperate line works the same way as the seperation on blank line 2. That being so that the code is easier to read for the coder?

Last edited 5 months ago by Mackenzie

0 Reply

Alex Author
Reply to Mackenzie 33 October 22, 2023 8:57 pm

C++ doesn't care where you put your curly braces. Some people like them at the end of the prior line (for conciseness), others (like me) prefer them on their own line (so they line up with the closing braces).

2 Reply

Kamil G.
Reply to Alex 34 January 31, 2024 6:51 am

I prefer it at the end of the prior line. It looks better after you minimize the long function for faster access to the code down below.

0 Reply

tall lightskin
October 8, 2023 2:05 pm

print('You are the GOAT.*Glazes softly*')

0 Reply

Sophia
September 21, 2023 8:42 pm

Well, i want to ask, in our first program Hello World, we use the library <iostream>, well, when we compile the program, we are compiling ALL the library codes or just the ones we're using? For example, if we're not using cin or endl, we're only using cout, it'll only compiles cout's code? Thanks for any answer.

3 Reply

Ben Hillen
Reply to Sophia 35 September 22, 2023 8:42 am

This is how a friend of mine described it

In C++, when you write a program, you often include libraries like <iostream> to use pre-written code for common tasks. When you compile your program, the compiler is smart enough to include only the specific parts of those libraries that your program actually uses. It doesn't include everything.

For example, if you're using cout to display output but not using cin to get input, the compiler will only include the necessary parts for displaying output. This keeps your program efficient and not "bloated" with unused code.

Think of it like ordering a pizza with various toppings. You only pay for and get the toppings you choose, not all the toppings available. Similarly, in C++, you only get the code you use from libraries.

So, don't worry about it; the compiler takes care of optimizing your program for you! As you continue learning, you'll become more familiar with how this process works.

I wish you the best of luck on you're c++ joinery

Last edited 6 months ago by Ben Hillen

7 Reply

arki
Reply to Ben Hillen 36 October 7, 2023 3:04 am

couldn't explain it any better. thanks! :)

0 Reply

Sophia

Reply to Ben Hillen 36 September 23, 2023 5:04 am

Okay thanks

0 Reply

Jfs

September 20, 2023 5:02 am

I get this error.

Untitled2.c|1|fatal error: iostream: No such file or directory

Using code::blocks btw

Last edited 6 months ago by Jfs

0 Reply

Kinuly

Reply to Jfs 37 September 20, 2023 7:09 pm

I believe the problem you are having is that your file is a .c which is used for creating c programs, whereas for c++ programs you should use .cpp.

1 Reply

Patricia

September 8, 2023 4:48 pm

Excellent and didactic way to teach C++. I apprenciate it.

3 Reply

Abdullah

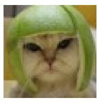
August 15, 2023 4:20 pm

While learning these great lessons I write notes to note any key points. My question is, if it would make sense to write, "The `std::cout` (stands for "character output") and the `<<` operator allows us to send information that can be displayed onto the console through characters, like the string `'Hello world'`."? Because I thought it wasn't clear enough:

Line 5... std::cout (which stands for “character output”) and the << operator allow us to send letters or numbers to the console to be output. In this case, we’re sending it the text “Hello world!”, which will be output to the console. This statement creates the visible output of the program.

I was suggesting If it could be made clearer (idk if I even make sense but yeah).

2 Reply


Alex Author
Reply to Abdullah 38 August 17, 2023 6:08 pm

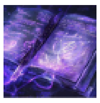
Rewrote as, " `std::cout` (which stands for “character output”) and the `<<` operator allow us to display information on the console. In this case, we’re displaying the text “Hello world!”. This statement creates the visible output of the program."

4 Reply

D O
August 4, 2023 10:35 am

If I'm learning to code at 13(14 in like 2 weeks) am I too late?

1 Reply


Kamil G.
Reply to D O 39 January 31, 2024 7:22 am

Never is too late for things like learning, whether it's coding or anything else. You can start learning something or go to college even in your nineties+, because we are learning our whole life.

The essence of life itself is learning and changing nonstop, all the way to death. Only when we are dead do we stop learning and changing.

The mindset of thinking that you are too late for something is only slowing you down in this process of infinite learning, making it harder and giving you an excuse to quit when you should persevere.

0 Reply

Carl
Reply to D O 39 November 8, 2023 9:08 pm

Well dang, if learning to code at 13.5 is too late, what does that make me, switching careers into programming at 34?

1 Reply

Sigh

Reply to D O ^{39} October 11, 2023 1:41 pm

I'm a cs student who just switched from mechanical engineering. I wish I had committed back when I was your age. You are gettign into it at the perfect time. That said, I am still learning will do just fine in CS. You can start whenever, even in college.

1 Reply

Undead34

Reply to D O ^{39} August 18, 2023 10:24 am

Ha, ha, not late, I think it's too early, you're very smart, I started at 16.

1 Reply

Anon

Reply to D O ^{39} August 13, 2023 7:25 pm

I won't be answering your question but I will give you an advice.

Remove that mindset of yours that tells you that you're too late for something. Especially learning.
No such thing as too late to learn. I first learned how to build a bike before I learned how to ride one at 20.

5 Reply

Anon

Reply to D O ^{39} August 7, 2023 11:22 am

"If I'm learning to code at 13(14 in like 2 weeks) am I too late?"

Brother if I'm learning to code at 19 am I too late?

if you're telling me you feel like you're too late
then I'm already dead and buried 5 meters deep down into earth

9 Reply

arki

Reply to Anon ^{40} October 7, 2023 3:08 am

haha!

1 Reply

Alex Author
Reply to D O 39 August 6, 2023 7:20 pm

Not too late by any means.

4 Reply

Masacre
August 1, 2023 11:15 am

nice

1 Reply

N T
July 25, 2023 4:34 am

> A computer program is a sequence of instructions that tell the computer what to do. A statement is a type of instruction that causes the program to perform some action.

I understand that this is saying that a statement is just one type of instruction, but would it be possible to clarify difference between an instruction (telling the computer what to do) and a statement (causing the program to perform some action)? I'm not sure I understand the difference between "telling the computer what to do" and "causing the program to perform some action".

1 Reply

Alex Author
Reply to N T 41 July 28, 2023 5:55 pm

"telling the computer what to do" and "causing the program to perform some action" are synonymous in this context.

The difference is that "instruction" is used informally to mean anything the computer can execute without being specific about what or how. It is intentionally vague to appeal to your common understanding.

A statement has a specific technical definition and syntax in the C++ language that the compiler understands.

As Kojoko notes, `#include <iostream>` is an instruction, as it tells the preprocessor to do something, but it's not a statement.

1 Reply

Kojoko

Reply to N T [41] July 28, 2023 9:07 am

For example, '#include <iostream>' isn't a statement since it doesn't tell the computer to perform any action, it just tells the compiler that we will be using this library.

0 Reply

Emeka Daniel

Reply to Kojoko [42] August 1, 2023 9:30 am

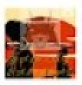
Yh more or less. A program is fundamentally just a bunch of instructions -might be related or unrelated- that tells your computer what to do, functions are now parts of programs that contain specific instructions for your computer. In a broader sense programs are collections of functions while functions are collections of instructions, then statements are the individual instructions themselves.

->For example, '#include <iostream>' isn't a statement since it doesn't tell the computer to perform any action, it just tells the compiler that we will be using this library.

The #include preprocessor directive is an instruction nonetheless, but not one addressed to your computer but to a program called a preprocessor [like the Author pointed out].

Last edited 8 months ago by Emeka Daniel

0 Reply



SID

Reply to Emeka Daniel [43] April 3, 2024 3:30 am

what I understand is that some instructions like "#include<iostream>" is directed to the computer or operating system and there are also instructions that is directed to the computer or operating system but it needs the compiler for that direction to change it to the machine language so that the computer can understand. So even though "return 0" of the main function is directed to the operating system it is dependent on the compiler to do so. Everything is an instruction 'for the computer (as it should be) but when it is dependent on the compiler it is a statement. please correct me if I am wrong I am new to programming

1 Reply

Emeka Daniel

Reply to SID [44] April 3, 2024 3:55 am

Yh, exactly. To your computer, everything is just an instruction that needs execution, but high level languages like C++ introduce semantic concepts like expressions, statements and expression statements to abstract and provide concrete definitions for the low level details.

i am new to programming : awesome, wishing you the best.

0 Reply

Krishnakumar
Reply to Emeka Daniel [43] November 14, 2023 7:49 am

> not one addressed to your computer

not addressed to the compiler you mean?

1 Reply

deng yunsheng
July 10, 2023 8:26 pm

thanks

0 Reply

Calin
June 24, 2023 3:50 am

What is the difference between std::cout and starting with using namespace std;

0 Reply

Yaetrna
Reply to Calin [45] June 24, 2023 7:45 am

`std::cout` only uses the `cout` function from the standard library, which is imported by `#include <iostream>`. If you use `namespace std;`, then you don't have to write the `std::` in front of `cout` anymore, because by `namespace std;` you tell the compiler that you use the whole standard library in your code. But this is not recommended Why? (https://stackoverflow.com/questions/1452721/why-is-using-namespace-std-considered-bad-practice)[46].

0 Reply

Porcupine
June 12, 2023 12:39 pm

I don't see anything pop out of the output box when I remove the;(semicolon). I think it's one of the setting I change like warning or other change setting. When I go to the build option I only see one option which is some thing analyze solution.

0 Reply

Noname
May 29, 2023 12:02 am

```
#include<iostream>

int main()
{
std::cout<<"First day">>;
return 0;
}
```

5 Reply

Links

1. https://www.learncpp.com/author/Alex/
2. https://computerehub.com/humix/video/BgnlmzWr7Bf
3. https://www.learncpp.com/cpp-tutorial/a-few-common-cpp-problems/
4. javascript:void(0)
5. https://www.learncpp.com/cpp-tutorial/introduction-to-the-compiler-linker-and-libraries/
6. https://www.learncpp.com/cpp-tutorial/comments/
7. https://www.learncpp.com/
8. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-choosing-a-language-standard/
9. https://www.learncpp.com/statements-and-the-structure-of-a-program/
10. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/
11. https://www.learncpp.com/wordpress/tiga-102-fix-for-wordpress-22/
12. https://gravatar.com/
13. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-595454
14. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-595465
15. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-594011

16. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-594101
17. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-594113
18. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-594125
19. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-593495
20. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-593554
21. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-593133
22. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-592653
23. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-592654
24. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-591870
25. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-591879
26. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-591635
27. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-591639
28. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-591403
29. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-590901
30. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-590186
31. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-589749
32. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-589545
33. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-589023
34. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-589026
35. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-587570
36. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-587617
37. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-587460
38. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-585619

39. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-585086
40. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-585240
41. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-584461
42. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-584659
43. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-584884
44. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-595390
45. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/#comment-582419
46. https://stackoverflow.com/questions/1452721/why-is-using-namespace-std-considered-bad-practice